5-31-2011

802.15.4/ZigBee Analysis and Security: tools for practical exploration of the attack surface

Ricky A. Melgares
Dartmouth College

Dartmouth Computer Science Technical Report TR2011-689

802.15.4/ZigBee Analysis and Security: Tools for practical exploration of the attack surface

Ricky A. Melgares

advised by Sergey Bratus

Computer Science Department

Dartmouth College

Hanover, New Hampshire

ricky.a.melgares.11@alum.dartmouth.org

May 31, 2011

Abstract

This thesis explores methods and techniques for surveying 802.15.4 and ZigBee wireless networks. The tools developed will aid in reconnaissance attacks against target networks; information gathered during this process will be used to profile a target network and its devices, as well as to pinpoint the geolocation of devices for executing physical attacks against the onboard hardware. Attacks against the PHY and MAC layers of the 802.15.4 standard will be explored as well.

Contents

1 Introduction

For the last decade or so, we have seen a proliferation of wireless technologies such as WiFi, Bluetooth, WiMax, and Wireless USB, just to name a few. Bluetooth, however, was the first to emerge as a WPAN (wireless personal area network), a short-range wireless technology that allows devices in close proximity of each other to exchange information [1]. The IEEE 802.15.1 specification subsequently emerged as a derivative standard of Bluetooth, giving rise to the IEEE 802.15 working group specializing in WPAN standards, responsible for the IEEE 802.15.4 standard subject of this thesis. The 802.15.4 standard defines the PHY and MAC sublayer specifications for LR-WPAN, aimed at low-rate, low-power, low-complexity, and low-cost wireless connectivity [2].

1.1 What is ZigBee?

ZigBee is a wireless communications standard that uses 802.15.4 as a basis for the PHY and MAC layers, while providing network functionality and an application framework at the higher layers [3]. ZigBee reinforces the low-rate and low-power characteristics of 802.15.4, and extends it by providing networking, routing, reliability, security, and a variety of application profiles for inter-operability between different vendor implementations [4].

1.2 How is ZigBee used?

The use of 802.15.4 and ZigBee in the home automation, industrial control, health care, and other industries is proliferating. CenterPoint Energy selected Itron as its smart meter provider in 2008, which provided smart meters with built-in ZigBee radio chips for a "built-in communications pathway into the home for data presentation, load control and demand response" [5]. Government stimulus and mandates have also pushed public utility companies towards the adoption of 802.15.4 and ZigBee for their smart energy needs [6]. In the home automation market, Yale, a leading lock manufacturer, has slated a 2011 product launch of "smart" deadbolts for integration into home security and automation systems such as those offered by Control4, a leading home automation solutions provider which also uses ZigBee in some of its products [7]. In the industrial and commercial setting, 802.15.4 is being used for remote control and monitoring, usually as an extension or for integra-

tion into existing SCADA[1] or BACnet[2] systems. At Dartmouth College for example, manual meter reading has been discontinued in favor of a wireless metering network that makes use of wireless sensors and repeaters/routers utilizing 802.15.4, for metering condensate, chilled water, and electricity usage as part of the campus energy management system [8]. The switch to wireless metering by the College was partly funded by a grant from the State of New Hampshire Greenhouse Gas Emissions Reduction Fund [9], which is an example of how state mandates and funding are helping to encourage the adoption of smart metering technologies such as 802.15.4 and ZigBee.

1.3 Problem Statement

ZigBee has yet to achieve widespread adoption, and thus the state of available wireless tools for the assessment of 802.15.4 and ZigBee can be compared to the early years of 802.11, which were dominated by a lack of available tools for the assessment of 802.11 security. Now, fourteen years after the release of 802.11a, one can find dozens if not hundreds of wireless tools and frameworks available for the assessment of 802.11. A handful of these tools are now regarded as the de facto standard when it comes to performing 802.11 wireless assessments, some of which include: Kismet, an 802.11 wireless network detector, sniffer, and intrusion detection system[11]; Aircrack-ng, an 802.11 WEP and WPA-PSK key cracking program[12]; Airbase-ng, a multi-purpose tool aimed at attacking wireless clients[13]; and Airpwn, a framework for 802.11 packet injection[14]. Currently no de facto standard exists for performing 802.15.4 and ZigBee wireless assessments, except for KillerBee, which is quickly taking the lead as the tool of choice for exploring and performing assessments of 802.15.4 and ZigBee networks and security research. Additionally, available commercial tools are expensive, which limits their use to vendors whose primary concern may not be the security of their products, and who typically use these tools for debugging protocol implementations rather than for security assessments. As ZigBee continues to gain momentum, it is important that free and open-source tools be available for surveying and evaluating the security of 802.15.4 and ZigBee wireless networks, which is one of the things that this thesis aims to improve, by lowering the barrier of entry for all types of operations with a limited budget for self-assessment and security research.

[1] Supervisory control and data acquisition system

[2] Data communications standard for building automation and control networks

Why is it important? Developing surveying tools and evaluating the security of the ZigBee wireless protocol is important, as ZigBee touches the kinetic framework[3] of systems more so than other wireless technologies. Since many ZigBee and 802.15.4 devices interface the physical environment to real-world applications, this makes them an attractive target to malicious actors bent on causing physical harm or disruption. Additionally, as ZigBee continuous to gain momentum, organizations are going to jump on the ZigBee bandwagon without truly understanding or assessing the security of the technology they are adopting.

1.4 Structure of Thesis

Section 2 discusses the current state of ZigBee security research and tools. Section 3 covers techniques and tools developed for surveying 802.15.4 and ZigBee wireless networks. Section 4 discusses the exploitation of ZigBee security, by breaking confidentiality and attacking the link layer. Finally, a summary and conclusion is presented at the end.

[3]Physical manipulation or control of switches, valves, etc.

2 State of the Art

While available ZigBee research and tools is sparse, there are a handful of security researchers at the forefront of ZigBee security research who have made their work and tools open-source and available.

KillerBee

Joshua Wright recently made available a Python-based framework codenamed KillerBee, aimed as an attack framework for exploiting 802.15.4 and ZigBee security [15]. The KillerBee framework provides facilities for discovering ZigBee networks, capturing and injecting traffic, as well as some rudimentary device locationing. The goal of the project is to provide users with a framework that can be extended and used as base for building other tools, discovering weaknesses, and carrying out a variety of attacks. My extensions and modifications to the KillerBee framework are discussed in Section 3.

Travis Goodspeed

Travis Goodspeed is best known for his work on attacking the software and hardware cryptosystems of microcontrollers and ZigBee radios. He has developed techniques for the extraction of encryption keys from protected memory, which coupled with Joshua's KillerBee framework, can be used to decrypt captured and encrypted traffic [16]. Travis also presented a PRNG vulnerability in TI's Z-Stack ZigBee implementation, which allowed for the remote recovery of keys [17]. Travis is also the creator and maintainer of the GoodFET, an open-source JTAG adapter, which can be used as an aid in key extraction and for debugging live chips [18].

Others

Kevin Finisterre recently built a ZigBee wardriving rig consisting of 16 ZigBee sniffing devices, whom I worked with on debugging USB bus issues when using KillerBee. Other researchers in the academic setting have released papers discussing theoretical attacks for sensor networks in general, with a focus on other protocols such as 6LoWPAN, and therefore a lack in ZigBee specific material. Furthermore, accompanying tools and software demoed or talked about in academic papers have not been made open-source or easily available, which is not in accordance with the aim of this thesis to address the lack of freely available and open-source tools. The author made various attempts at acquiring relevant software and code from various academic researchers

to no avail. Additionally, academic researchers will continue to reinvent the wheel over and over again until a de facto standard or easily available open-source tool becomes available for exploring the attack surface, where the attack surface represents the exposure or potential risk posed by both known and unknown vulnerabilities or security flaws. Only then can the security of 802.15.4/ZigBee network be truly assessed and put through its paces.

Collaboration

I have been in touch with Joshua Wright, Travis Goodspeed, Michael Ossmann, and Kevin Finisterre among other security researchers, and have sought advice regarding certain aspects of this thesis, and additionally, have been collaborating with them on several issues and goals by sharing information, tools, and techniques. This collaboration has resulted in the merging and integration of code developed as part of this thesis into open-source projects. I have also been working jointly and in parallel with Ryan Speers in developing tools and techniques for the practical exploration of 802.15.4 and ZigBee security.

3 Identifying 802.15.4 and Zigbee Networks and Devices

Identifying 802.15.4/ZigBee networks is of particular importance because of the distributed nature and mesh topology of 802.15.4/ZigBee networks. Traditional wardriving methods are less effective when applied to 802.15.4/ZigBee networks, because of the short-range and "quiet" nature of these devices. One must therefore tailor traditional site surveying and wardriving techniques for 802.15.4/ZigBee networks. This section will discuss improvements to traditional techniques, and the importance of analyzing network traffic and pinpointing the physical location of devices, as well as selection of hardware.

3.1 Equipment

A variety of hardware equipment was acquired for the purposes of this thesis. Each one was evaluated and modified, both in terms of hardware and software.

RZUSBstick The initial KillerBee framework provided support solely for the RZUSBstick from ATMEL. Additionally, the RZUSBsticks acquired were flashed with firmware provided by Joshua Wright, which adds injection capabilities. The RZUSB is an attractive hardware platform because of its low cost ($40), sniffing/injection capabilities, and extended support from ATMEL, which provide extensive documentation and facilities for modifying the device firmware, making it a flexible and hacker-friendly platform. The internal PCB antenna, however, proved to be less than ideal for wardriving during initial tests. I therefore researched and acquired other hardware platforms with external antenna capabilities for capturing network and device traffic from longer distances, an important capability to have when wardriving discreetly and from a vehicle for example.

ZENA Network Analyzer The ZENA Network Analyzer by Microchip was designated solely as a sniffing platform for the purposes of this thesis. The closed nature of the software platform on the ZENA prevents one from adding injection capabilities or making any modifications to the firmware. The ZENA is therefore better suited for capturing and analyzing network traffic. The ZENA was selected primarily for its external antenna support, a capability which I had to enable by relocating a capacitor

on the board to direct the trace coming from MRF24J40 radio to an external antenna trace, a feature that is not documented by Microchip and which they failed to provide information about when responding to my inquiries regarding the capability over the phone. After relocating the capacitor, an RP-SMA PCB mount was soldered, allowing one to use an external and higher-gain RP-SMA antenna of choice, pictured in Figure 1.

After the hardware modifications were made, RF tests using a directional Yagi antenna yielded significant improvements in site surveying and wardriving results.

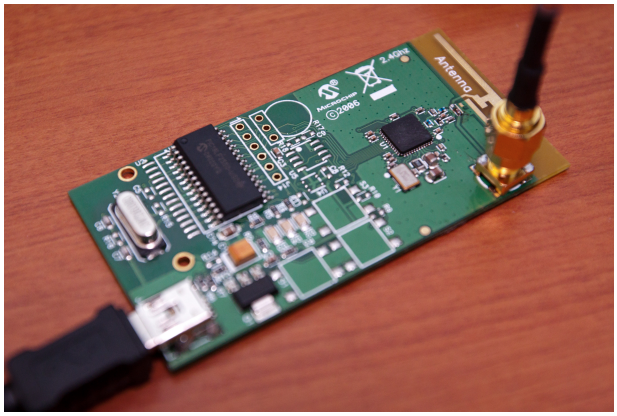

Figure 1: ZENA modifications

I also developed a software driver for the ZENA for use with the KillerBee framework. Although Joshua Wright had written some basic Python code to dump raw packets from the ZENA to a terminal over serial, support for the ZENA was not available in KillerBee. I therefore used Joshua Wright's code as a starting point and incorporated support for the ZENA into KillerBee, allowing one to use the KillerBee framework and ZENA for long-range traffic capture.

Tmote I also evaluated the Tmote Sky hardware platform in the early stages of the thesis. The Tmote Sky is an open-source mote platform containing an MSP430 microcontroller and 802.15.4 compliant radio. I acquired the Tmote Sky platform in large quantities from the Wireless Sensor Lab at Dartmouth, which made the platform suitable for use as target devices and networks in addition to using the device as a network and traffic analysis platform. The Tmote Sky also provided the facility to solder an RP-SMA mount for external antenna support, in much the same manner as the ZENA, and therefore surpassed the ZENA as the network and traffic analysis platform of choice.

To designate the Tmote as a sniffing platform, I developed code to sniff traffic on a given channel and dump raw packets to the terminal over serial. I initially used the TinyOS operating system and framework to develop the sniffing application, which later proved to be a nuisance when trying to

add more advanced functionality such as threading and client commands. I therefore decided to abandon the use of the TinyOS framework. Fortunately and after this decision was made, Travis Goodspeed ported a version of the GoodFET project to the Tmote/ TelosB platform, providing sniffing and injection capabilities using low level access to the hardware. I have also contributed to the GoodFET project through testing and code development of the GoodFET project since the introduction of the Tmote/TelosB port by Goodspeed.

3.2 Capturing and Analyzing Network Traffic

To analyze the security of ZigBee networks, the facilities for discovering networks and analyzing traffic must be in place. I therefore developed an all-channel monitor application for integration with the KillerBee framework. The need for an all-channel monitor came into being when attempting to capture traffic and reverse engineer the operation of the Spinwave Wireless Sensor Network devices in use by Dartmouth College's Facilities Operations and Management (FO&M), as part of the campus energy management system. Spinwave devices were acquired with permission from an FO&M employee to build a test network in a lab environment, to avoid interfering with the production sensor network in use by the College.

When attempting to capture traffic from the test Spinwave network, I noticed that the same PANID of the target network would appear on different channels, which is not standard 802.15.4/ZigBee behavior. After doing some research, it was determined from the documentation that the Spinwave devices use a proprietary channel-hopping algorithm that continuously switches between the 4 best channels, from the available 16 IEEE 802.15.4 channels, for reliability and interference avoidance in overlapping Bluetooth and 802.11 channels, since they share part of the 2.4 GHz spectrum[10].

3.2.1 OpenEar

OpenEar is a 16-channel monitor that draws on the KillerBee framework by assigning a channel to each device attached, meaning that 16 devices are required for simultaneous monitoring of all channels. OpenEar also extends the KillerBee framework with user-space threading of the capture process, so that a single program is able to initialize and shut down captures for all channels at once. This differs from Kevin Finisterre's efforts, since his method of building an all-channel monitor involved scripting the start-up of a single non-threaded process for each device and channel using a BASH script. This also differs from Ryan Speer's `zbwardrive` tool, whose purpose is to efficiently allocate and use a limited number of devices for capturing traffic at any given time in an area, where a small number of networks may be present and where channel-hopping is not expected.

The OpenEar application therefore enumerates all of the attached devices, assigns a channel number to each, initializes a packet capture file for each channel and a database connection for logging information to a MySQL server using a database schema designed and provided by Ryan Speers. Additionally, I created a persistent in-terminal display for displaying real-time

stats, which is an important feature for performing real-time analysis and identifying "chatty" or saturated areas of service when wardriving or on site. The OpenEar application proved to be useful when mapping and gathering information about 802.15.4 wireless coverage on the Dartmouth Campus.

Threading an all-channel capture process also allows for the sharing of resources between capture processes, such as location information from an attached GPS device, a feature which is discussed in the next section.

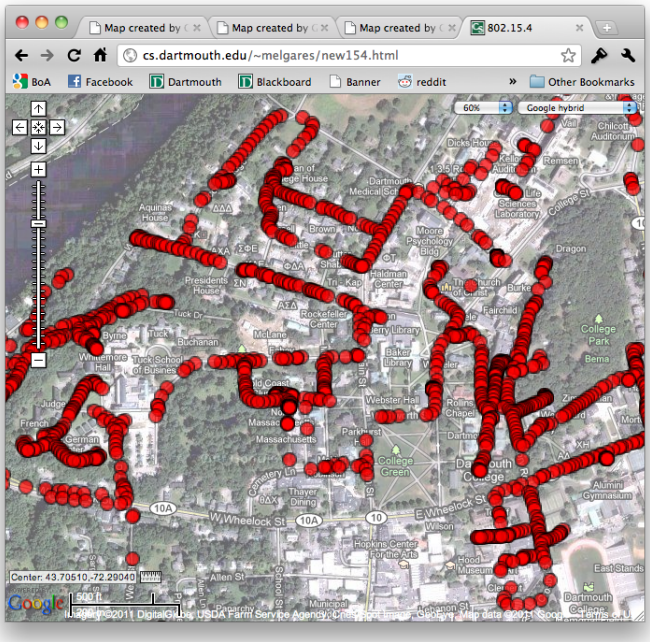

Figure 2: 802.15.4 packet plot

3.2.2 Location Logging and Estimation

While KillerBee provides a location estimation tool, it is not enough for efficiently locating devices, as it simply acts as an RSSI strength meter, which has proven to be an inefficient method for locating devices in site surveying and wardriving tests conducted. I therefore extended the KillerBee framework by adding the ability to log location information along with captured packets. To enable the logging of location information, I used gpsd, an open-source daemon that takes data received from an attached GPS device, and formats the GPS data into a JSON format and gives back the GPS data over a local TCP/IP connection. This allows multiple applications to share and use a GPS device concurrently. This is necessary as most GPS devices capable of outputting GPS data to a computer do so over a serial connection, which cannot be shared or accessed by multiple applications at once. Gpsd additionally provides a Python API, which facilitates the integration of gpsd into the KillerBee framework, which is also written in Python. The OpenEar application therefore starts a "lo-

cation" thread in addition to the capture threads. The sole purpose of the "location" thread is to continuously poll the gpsd daemon service and update the current location (latitude, longitude, altitude), referenced each time a packet is logged. Currently location information is logged to a database only. Future work will therefore incorporate logging of location data into the PCAP files as well by utilizing Johnny Cache's PPI-Geolocation standard, which uses per-packet information tags. Support for the PPI-Geolocation standard has been worked into Wireshark, Scapy, and Kismet, making it ideal for standardizing the logging of location information across a variety of capture applications, rather than having to come up with a non-standardized format of storing location information along with packets.

Logging location information is important for post-analysis and geolocationing. One can use location information to identify saturated areas of service or to pinpoint the exact location of a device or network, information which can then be used to profile a network and for carrying out physical attacks against the hardware (discussed in the next section). Figure 2 shows a simple plot 802.15.4 traffic across the Dartmouth Campus, where each circle represents a point where an 802.15.4 packet was logged. A more advanced geospatial analysis takes into consideration the received strength and grouping of packets by network or relationship. I experimented with the use of GIS applications such as ArcMap for these purposes, which provide the ability to perform advanced spatial interpolation, an example of which is shown in Figure 3, which uses the hotspot analysis interpolation tool in ArcMap to interpolate and estimate wireless coverage across the campus based on the limited number of sample points and strength readings taken. I have developed and written scripts for extracting location data from the database and converting into the shapefile format, which is a geospatial vector data format that can be imported into GIS applications

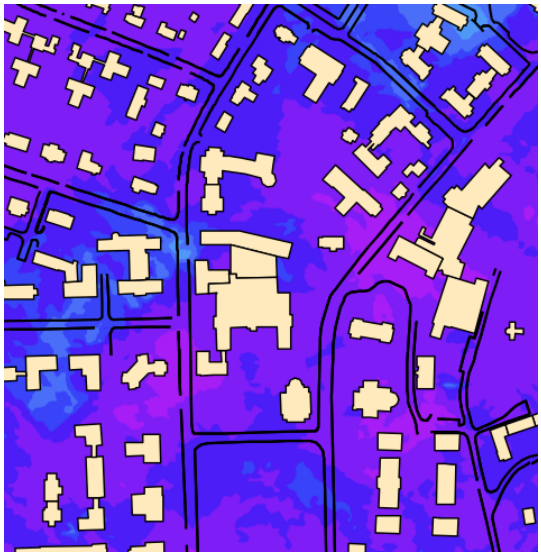

Figure 3: ArcMap spatial interpolation using the hotspot analysis tool

such as ArcMap. The written csv2shp.py program takes in a csv file of the the form "ID, latitude, longitude, RSSI", where ID is either the name of the network or device and RSSI is the received signal strength, and creates a shapefile using the open-source GDAL (Geospatial Data Abstraction Library) framework, which includes Python bindings.

The importance of pinpointing the location of a device is discussed in the next section.

4 Attacking the Link Layer

For the purposes of evaluating 802.15.4/ZigBee security, I explored the development of a tools and techniques that can used as a pivot for launching a variety of attacks, primarily using the USRP (Universal Software Radio Peripheral) and GNURadio framework[20]. Specifically, I explored the means of attacking the link layer, which handles physical access to the wireless medium, by using RF (radio frequency) jamming techniques to intercept and corrupt traffic. Furthermore, the application of location knowledge of the target is applied as a means of breaking confidentiality.

4.1 USRP

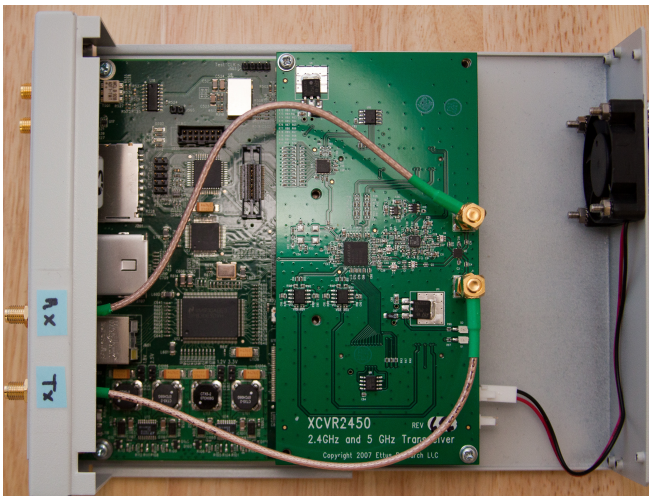

Figure 4: USRP2 and 2.4Ghz daughterboard

One of the other hardware platforms that was used during this thesis is the USRP2 developed by Ettus Research. The USRP is essentially an FPGA with swappable daughterboard cards that act as the RF front end for the device, pictured in Figure 4. Components typically implemented in hardware, such as amplifiers, demodulators, etc., are instead implemented in software. This means that a minimal amount of data processing and filtering is actually performed on the USRP or FPGA itself. It is instead left up to the host computer to perform filtering and digital signal processing, which will generally have a significantly higher computing power than the FPGA. The USRP is therefore a flexible platform which the operation of can be changed easily and on the fly, since signal processing blocks are written and implemented in software instead of hardware. This makes the USRP an RF Swiss Army knife of sorts when coupled with an array of daughterboards cards that enable the reception and transmission in anywhere from the 25 MHz to 5 GHz range with the appropriate daughterboard cards.

802.15.4 on the USRP

The USRP2 and GNURadio framework were coupled with Leslie Choong's UCLA 802.15.4 framework[21], which is an implementation of the 802.15.4 specification using the GNURadio framework, for enabling the reception and transmission of 802.15.4 traffic on the USRP. Originally written for the USRP1, I ported the framework for use with the USRP2 and made modifications to the framework to enable the ability to target and jam 802.15.4 packets, which is discussed in the next section.

4.2 Jamming

Selective jamming is the process of listening for a specific transmission or packet and disrupting it by transmitting garbage or a packet at the same time. Although this technique has been covered numerous times by academic researchers in the past[23], the aim is to lower the barrier of entry by implementing the selective jamming technique on a platform that is commonly used by security researchers.

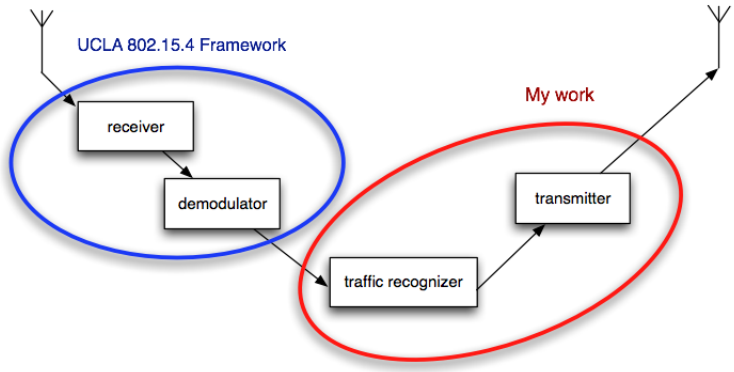

Figure 5: USRP2 Selective Jamming Architecture

4.2.1 Why jam selectively?

Selective jamming differs from continuos jamming which is the more common way of jamming. Selective jamming has two main advantages over continuos jamming, the first being that the jammer is active for a less period of time and therefore consumes less energy, which is important if you're running off batteries for example. Secondly, it allows the person doing the jamming to operate covertly. It will not be obvious to the network operator or owner that someone is jamming their network. The jamming transmission needs to only be activated for a period for a short period of time after the target

transmission or packet has been identified, long enough to flip a couple of bits in the packet. This is useful because most spectrum analyzers are not going to have a fine enough resolution to to see a microsecond or even millisecond blip, unless by chance. This is because spectrum analyzers have to sweep the frequency range, reporting what they say at each frequency along the way, a process which can take a couple hundred milliseconds at best to sweep across the entire range. I experienced this first hand when trying to debug the selective jamming application on the USRP2 using the WiSpy, a USB spectrum analyzer, which has a resolution to hundredths of a second.

4.2.2 What are some targets for selective jam?

Targets for selective jamming include: ACKs, jamming ACKs will cause congestion on the network because packets have to be re-transmitted (Figure 6 shows an example of a jammed ACK; association responses, jamming these will cause a denial of service and power drain on battery operated devices since they cannot join the network and thus have to continuously retry to join the network, an attack that the I carried out using the selective jamming architecture developed for the USRP2.

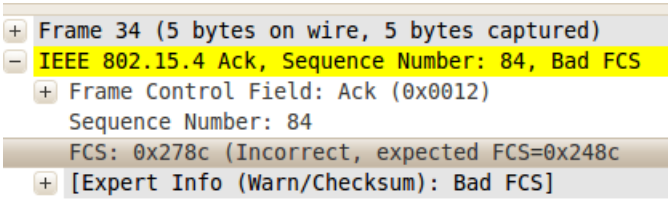

Figure 6: Selectively Jammed ACK packet

4.2.3 Implementation

To implement selective jamming on the USRP2, I implemented a traffic recognizer and jammer/transmitter component blocks into the UCLA framework. Figure 5 shows the selective jamming architecture. The receiver and demodulator components are provided/given by the UCLA framework, whereas the traffic recognizer and transmitter components represent code written by me. Given a mask to match against, the traffic recognizer reads the incoming packet bytes one by one, comparing the bytes against the mask. Upon a match, the transmitter is activated, which simply generates noise on the given channel, although a random packet could be sent as well.

4.2.4 Limitations

One limitation encountered while working with the USRP2 was being able to analyze packets in real time so that the transmission can be disrupted before it finishes transmitting. In all of the USRP2's glory, the latency between the USRP and host computer prevents one from jamming packets in real-time or within the same frame. Despite this limitation, the USRP2 was able to inference and jam particularly long sequences or exchanges, such as that occurs when a device joins a network. Similarly, we one can check to see if the ACK bit for a particular packet is set, and if so, will indicate that an ACK is expected as the next packet, meaning that we can inference and activate our jammer ahead of time.

The GNURadio and UCLA framework are also written in Python, which is not necessarily ideal for real-time processing. One improvement that could be made is to move the traffic recognizer component to the FPGA, so that the FPGA is able to match a packet or transmission locally without having the host computer to do the matching/parsing. Doing so requires modifying the firmware on the FPGA, written in VHDL, for which I did not get a chance to do due to time limitations.

This type of attack is therefore better suited either with modified FPGA firmware or a microcontroller, as demonstrated by Ryan Speers reactive jamming implementation on the Tmote, which was pursued after it became clear that the USRP2 would simply not be fast enough for achieving a fast enough turnaround for jamming.

5 Breaking Confidentiality and Integrity

Direction finding in 802.15.4 and ZigBee networks is of particular importance because of the distributed nature and mesh topology of 802.15.4/ZigBee networks. One of the attacks that one can carry with location information is physical capture or tampering. Travis Goodspeed has done some work on this, including the development of the GoodFET, which can be used as an aid in key extraction. Pictured in Figure 7 is a production in-home display (IDH) that I acquired and extracted keying information from, which included certificates, in addition to firmware and encryptions keys. The IDH contains an EM250, which contains no security fuses or protective measures in place for protecting the memory. I was thus able to extract certificate and key information from the device with relative ease using vendor tools. The certificates extracted from the device could be used to forge the identity of an IDH. With physical capture one can also modify sensor inputs, either physically or in firmware, by replacing the firmware on the device with custom firmware.

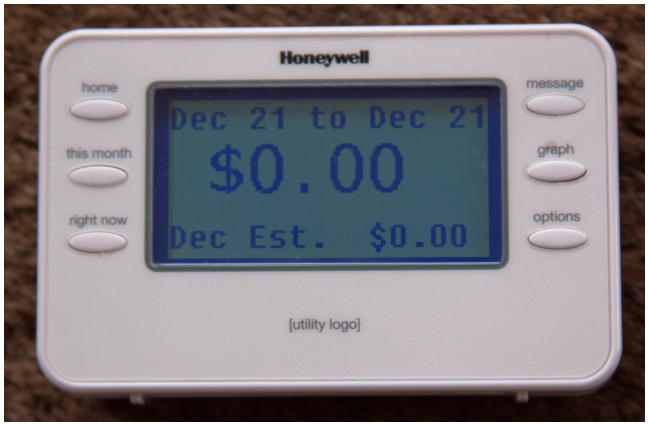

Figure 7: Production in-home display

5.1 Reverse Engineering the IDH

In the process of reverse engineering the device, the first thing to do was to look up the FCC documentation for the device. In typical FCC vendor submission style, the interesting parts of the internal photos were blurred so as to hide information about the particular radio chip being used.

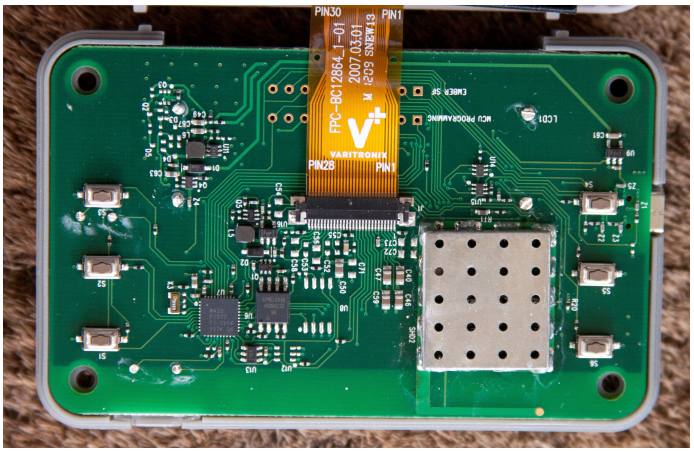

Figure 8: In-home display internals w/ shield

The next was therefore to open the IDH and examine the hardware. Upon doing so, I discovered that the radio part of the IDH had a shield soldered. Whether the purpose of the shield was to shield the radio from RF interference or to provide some sort of physical security remains unclear, in either case, I was able to desolder the shield after 15 minutes of work. Upon removing the shield, I discovered that the IDH was using an off-the-shelf EM250 radio module, pictured in Figure 9.

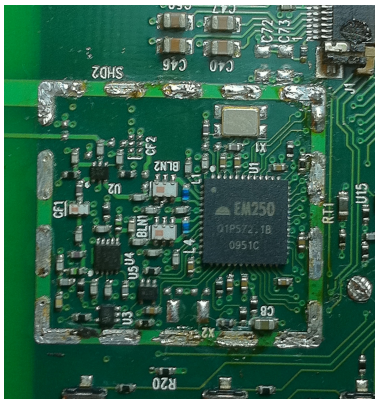

Figure 9: In-home display internals w/o shield

After the type of radio chip being used was determined, the next step was to lookup up manufacturer documentation to see how one might go about extracting/dumping the firmware.

It was determined from the documentation that the InSight USB Link, a stand-alone USB programmer for programming the EM250 and EM260 devices, would be

sufficient for not only programming the device but also extracting memory and dumping the radio chip's firmware. Once the debugger was acquired, I used Ember tools and software to dump the device firmware as a hex file. To do this, the author connected the debugger to the SIF port of the IDH, which provides and access to the entire flash and RAM memory space, and used the `EM2USBRead` utility to dump the chip's entire flash memory using the following command: `$ ./EM2USBRead.exe -Run -outfile dump.hex`. This command reads the entire chip's flash contents, including application data, boot loader, and manufacturing tokens, where information such as certificates are typically placed, enabling easy access to security tokens such as private keys and certificates.

More specifically, four security components can be extracted from the EM250. The first component is the device's IEEE address and static public key signed with the CA's private key, and thus confidentiality is not significantly impacted. The static private key on the other hand is generally carefully guarded and treated as secret information, and significantly impacts confidentiality. The CA public key corresponds to the secret private key held by Certicom, and is used to authenticate other devices' certificates and to validate encryption keys generated using the Smart Energy Key Establishment Cluster, and is also regarded as public information, and thus confidentiality of the CA public key not of importance.

Although not necessarily a new or novel attack against the crypto system, the capability to extract key information memory shows a disturbing trend, where vendors either continue to remain ignorant or simply refuse to heed the advice of security researchers. Additionally, the security token information extracted from the device could be used to forge the identity of another device, which impacts integrity. In the case of the EM250, the EM260 would be better suited as it provides better protection through the use of security fuses to prevent the dumping of memory, although security may only be marginally better, as Travis Goodspeed has found ways of getting around security fuses on other radio chips and microcontrollers[22].

6 Conclusion

This thesis has covered the use of a range of tools, all of which have either been incorporated into the KillerBee framework or released individually as open-source code. Most of the research presented in this thesis was also presented at ShmooCon 2011 Washington, D.C. , where presented work garnered interest from various members in the security research community, many of which got in contact with the author shortly after the ShmooCon presentation. Other security researchers have thus been in contact to collaborate and pursue their own security research using the tools developed as part of this thesis.

Many different areas of research were touched upon as part of this thesis, everything from sniffing and carrying attacks using the highly customizable USRP2 and GNURadio framework, to locating and ultimately acquiring devices for physical extraction of information. Additionally, all research and tools have been made publicly available at: `http://www.cs.dartmouth.edu/~melgares/zigbee/`.

After spending some time working on researching and working on this thesis, it became clear to the author that much remains to be learned about 802.15.4/ZigBee security research, and that the state of 802.15.4/ZigBee security research and tools is evolving. It will therefore be interesting to see what new tools and research are released in the future.

7 Acknowledgments

The author wishes to thank Sergey Bratus, Ryan Speers, Joshua Wright, Travis Goodspeed, Michael Ossmann, and Kevin Finisterre for lending their expertise and support. This research was supported in part by the National Science Foundation under Grant Award Number 1016782 and the Department of Energy under Prime Award No. US DOE DE-OE0000097, Subaward No. 2010-01251-01. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NSF or DOE.

References

[1] The Bluetooth Special Interest Group, "Personal Area Networking Profile," www.bluetooth.com

[2] IEEE Comp. Soc. LAN/MAN Stds. Comm., New York, NY. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), Amendment 1: Add Alternate PHYs," IEEE Std 802.15.4a-2007.

[3] Gislason, Drew. *Zigbee Wireless Networking*, Newnes, Newton, MA, 2008.

[4] ZigBee Alliance. *ZigBee RF4CE Specification*, 1 edition, March 2009.

[5] Itron (May 14, 2008). "Itron Selected by CenterPoint Energy as AMI Technology Provider: Agreement Outlines Phased Deployment for OpenWay Meters in Houston." Press release. Retrieved 2010-12-3.

[6] Chaudhary, R. (2010, November 17). In-State: Smart Energy to Fuel 802.15.4 and ZigBee Adoption. *Smartgrid TMCNET*. Retrieved from http://smart-grid.tmcnet.com/topics/smart-grid/articles/118120-in-state-smart-energy-fuel-802154-zigbee-adoption.htm.

[7] Yale Locks (September 23, 2010). "Yale to Introduce Company's First Locks Designed to Integrate Seamlessly into Digital Home." Press release. Retrieved 2010-12-3.

[8] Dartmouth College FO&M. "Attachment 1: Campus Energy Management System."

[9] Spinwave Systems. "Project Profile - Wireless Metering Dartmouth College."

[10] Spinwave Systems. "User Manual - A3 Wireless Sensor Network."

[11] Kismet. `http://www.kismetwireless.net/`.

[12] Aircrack-ng. `http://www.aircrack-ng.org/`.

[13] Airbase-ng. `http://www.aircrack-ng.org/doku.php?id=airbase-ng`.

[14] Airpwn. http://airpwn.sourceforge.net/Airpwn.html.

[15] Joshua Wright, Killerbee: https://code.google.com/p/killerbee/.

[16] Travis Goodspeed, "Extracting Keys from SoC Zigbee Chips," http://travisgoodspeed.blogspot.com/2009/08/extracting-keys-from-soc-zigbee-chips.html

[17] Travis Goodspeed, "PRNG Vulnerability of Z-Stack ZigBee SEP ECC," http://travisgoodspeed.blogspot.com/2009/12/prng-vulnerability-of-z-stack-zigbee.html

[18] Travis Goodspeed, GoodFET, http://goodfet.sourceforge.net/.

[19] IEEE802.15.4/ZigBee Stack for Linux, http://sourceforge.net/apps/trac/linux-zigbee/wiki/WikiStart

[20] GNU Radio, http://gnuradio.org/redmine/wiki/gnuradio.

[21] Choong, Leslie, "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio," UCLA, 2009.

[22] Travis Goodspeed, "Extracting Keys from Second Generation Zigbee Chips," http://www.blackhat.com/presentations/bh-usa-09/GOODSPEED/BHUSA09-Goodspeed-ZigbeeChips-PAPER.pdf.

[23] Konings, B., Schaub, F., Kargl, F., Dietzel, S. "Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard." Proceedings of the IEEE 34th Conference on Local Computer Networks, LCN (2009)

[24] ATMEL, "AVR2002: Raven Radio Evaluation Software," http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4396.